USF UNIVERSITY OF SAN FRANCISCO

CHANGE THE WORLD FROM HERE

Inheritance

Generics

Chris Bryan ◆ cbryan2@usfca.edu
Department of Computer Science

Generic Types

- Enables types to be specified as a parameter when defining classes or methods
- Allows generalization of code, while still being able to provide some restrictions on type
- Reduces amount of casting that must happen in code

http://docs.oracle.com/javase/tutorial/java/generics/index.html

Collection Examples

```
1  // Using Generic types, ArrayList can be used
2  // with different types
3  ArrayList<String> a = new ArrayList<>();
4  ArrayList<Double> b = new ArrayList<>();
5
6  // Sometimes, we must specify multiple types
7  HashMap<String, String> c = new HashMap<>();
1  HashMap<String, ArrayList<Integer>> d = new HashMap<>();
```

http://docs.oracle.com/javase/tutorial/java/generics/index.html

Naming Convention

- Use a single uppercase letter to name a generic type
- Use E for an element (good default)
- Use K for a key element
- Use V for a value element
- Use N for a number element

Method Example

```
1  public <E> E chooseRandom(E item1, E item2) {
2      if (Math.random() > 0.5) {
3          return item1;
4      }
5      else {
6          return item2;
7      }
8  }
```

http://docs.oracle.com/javase/tutorial/java/generics/index.html

Class Example

```
1  public class Pair<K, V> {
2      private K key;
3      private V value;

4      public Pair(K key, V value) {
5          this.key = value;
6          this.value = value;
7      }
8  }
```

http://docs.oracle.com/javase/tutorial/java/generics/index.html

Comparable Example

```
1  public <B extends Comparable<B>> B
2          chooseMax(B item1, B item2) {
3      if (item1.compareTo(item2) > 0) {
4          return item1;
5      }
6      else {
7          return item2;
8      }
9  }
```

http://docs.oracle.com/javase/tutorial/java/generics/index.html

Wildcard Example

```
1  public double sumNumbers(
2          Collection<? extends Number> nums) {
3      double sum = 0.0;
4      for (Number n : nums) {
5          sum += n.doubleValue();
6      }
7      return sum;
8  }
```

http://docs.oracle.com/javase/tutorial/java/generics/index.html

What You Can Do

- Can declare one or more generic types when defining a method or a class
- Can restrict the generic type using bounding and inheritance relationships
- Can restrict the generic type using wildcards and upcasting references

http://docs.oracle.com/javase/tutorial/java/generics/index.html

What You Can NOT Do

- Cannot use primitive types as a generic type
- Cannot create an instance of a generic type
  - e.g. E elem = new E(); // error
- Cannot use generic types for static members
  - e.g. private static E elem; // error
- Other strange restrictions; see tutorial

http://docs.oracle.com/javase/tutorial/java/generics/index.html

Chris Bryan ◆ cbryan2@usfca.edu
Department of Computer Science